AMENDMENTS TO THE CLAIMS

Claims Pending:

- At time of the Action: Claims 1-38
- Amended Claims: Claims 1, 7, 15, 24, and 35
- After this Response: Claims 1-38

The following listing of claims replaces all prior versions and listings of claims in the application.

1. (Currently Amended) A method, comprising:

receiving an input, wherein the input comprises elemental language units;

evaluating the input against multiple queries by evaluating common query expressions of the multiple queries in parallel, at a same time;

generating at least some of the elemental language units into opcodes;

merging opcodes into an opcode tree, wherein no opcodes are added to the opcode tree during an active merging,

wherein the language units have been parsed and compiled into opcodes;

traversing the opcode tree of hierarchical nature that includes a plurality of opcode nodes which together define opcodes that should be executed to evaluate a plurality of queries;

executing each of the opcode nodes in the opcode tree as each opcode node is encountered in the traversal to evaluate the plurality of queries against the input;

indexing branch opcodes to provide a framework for insertion of indexing techniques that are customized to a type of comparison;

maintaining the opcode tree that is used during processing by making a copy of the opcode tree; and

updating the opcode tree copy, wherein the opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing;

wherein a relationship between the opcodes and the opcode tree is embedded in the opcodes that is created when a query is complied.

2. (Original) The method as recited in claim 1, the executing step further comprising using an intermediate result to execute an opcode node when the opcode node includes one or more ancestor opcode nodes that have been executed to derive the intermediate result.

3. (Original) The method as recited in claim 1, the executing step further comprising executing a single opcode node to evaluate at least a portion of at least two of the plurality of queries.

4. (Original) The method as recited in claim 1, the multiple queries further comprising XPath queries.

5. (Original) The method as recited in claim 1, the executing step further comprising executing a branch node to execute multiple opcode nodes that depend from the branch node, the branch node including an indexed branch lookup function.

6. (Original) The method as recited in claim 5, further comprising a hash function as the indexed branch lookup procedure.

7. (Currently Amended) An opcode tree data structure stored on one or more computer-readable media having computer executable instructions stored on a computing device, comprising a plurality of hierarchical opcode nodes that represent a plurality of opcodes that are executed as each opcode node is encountered to evaluate a set of queries represented by the opcode tree, wherein the opcode tree that is used during processing is copied and updated [[.]] ;

the instructions further comprising to evaluate an input against multiple queries by evaluating common query expressions of multiple queries in parallel at a same time; and

the instructions further comprising to update an opcode tree copy, wherein the plurality of opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing.

8. (Original) The opcode tree data structure as recited in claim 7, further comprising at least one branch node that includes an indexed lookup procedure that is executed to optimize execution of multiple opcode nodes that depend from the branch node.

9. (Original) The opcode tree data structure as recited in claim 8, the indexed lookup procedure further comprises an indexed lookup procedure selected from the following list of indexed lookup procedures: a hash table algorithm; an algorithm using tries; an interval tree algorithm.

10. (Original) The opcode tree data structure as recited in claim 7, wherein the queries further comprise XPath queries.

11. (Original) The opcode tree data structure as recited in claim 7, further comprising at least one shared segment that corresponds to multiple queries.

12. (Original) The opcode tree data structure as recited in claim 11, wherein a single execution of the shared segment evaluates at least a portion of each of the multiple queries.

13. (Original) An inverse query engine containing the opcode tree data structure as recited in claim 7.

14. (Original) The opcode tree data structure as recited in claim 7, further comprising a branch node that includes references to more than two dependent opcode nodes.

15. (Currently Amended) A query evaluation system, comprising:

memory;

a processor coupled to the memory for executing a query evaluation with elemental language;

a language analysis module generating elemental language input into opcodes, wherein input against multiple queries is evaluated by evaluating common query expressions of the multiple queries in parallel at a same time;;

an opcode merger configured to combine opcodes that are derived from compiling expressions into an opcode tree, wherein the opcode merger detects using an optimization algorithm and combines literal comparisons into an indexed literal branch opcodes, wherein no opcodes are added to the opcode tree during an active merging;

the opcode tree of hierarchical nature stored in memory and containing opcode nodes that include opcode objects corresponding to a plurality of queries, each opcode object that is common to multiple queries being represented by a single opcode node;

a query processor configured to execute each opcode node as encountered of the opcode tree one time to evaluate the plurality of queries; and

the opcode tree that is used during processing by the query processor is copied and updated, wherein the opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing;

wherein a relationship between the opcodes and the opcode tree is embedded in the opcodes that is created when a query is complied.

16. (Original) The query evaluation system as recited in claim 15, further comprising an input module configured to receive an input that is evaluated against each of the plurality of queries when the query processor executes the opcode nodes.

17. (Original) The query evaluation system as recited in claim 16 further configured to receive a SOAP (Simple Object Access Protocol) message as the input that is evaluated against the plurality of queries.

18. (Original) The query evaluation system as recited in claim 15, further comprising at least one branch node in the opcode tree that connects an opcode node to two or more dependent opcode nodes.

19. (Original) The query evaluation system as recited in claim 18, further comprising a branch node that includes a lookup routine to process the dependent opcode nodes.

20. (Original) The query evaluation system as recited in claim 15, further comprising an interim results cache that stores results of opcode node executions that are used in the execution of subsequent opcode nodes.

21. (Original) The query evaluation system as recited in claim 15, further comprising a filter table that stores the plurality of queries, the filter table further including a reference to the opcode tree.

22. (Original) The query evaluation system as recited in claim 15, an opcode object common to multiple queries further comprising an opcode object that is in a similar location of an opcode object sequence at the beginning of the multiple queries.

23. (Original) The query evaluation system as recited in claim 15 including queries that are XPath queries.

24. (Previously Presented) One or more computer-readable storage media containing computer-executable instructions that, when executed by a computer, perform the following steps:

evaluating input against multiple queries by evaluating common query expressions of the multiple queries in parallel at a same time;

generating an input of elemental language units into opcodes;

merging opcodes into an opcode tree, wherein the language units have been parsed and compiled into opcodes, wherein no opcodes are added to the opcode tree during an active merging;

executing opcode nodes as encountered in the opcode tree of hierarchical nature to evaluate a plurality of queries represented in the opcode tree, at least one opcode node corresponding to at least a portion of two or more of the plurality of queries;

indexing branch opcodes to provide a framework for insertion of indexing techniques that are customized to a type of comparison;

caching an execution context derived from the execution of a first segment of opcode nodes;

re-using the execution context when executing a second opcode node;

maintaining the opcode tree that is used during processing by making a copy of the opcode tree; and

updating the opcode tree copy, wherein the opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing;

wherein a relationship between the opcodes and the opcode tree is embedded in the opcodes that is created when a query is complied.

25. (Previously Presented) The one or more computer-readable storage media as recited in claim 24, wherein the first segment of opcode nodes includes ancestor opcode nodes of the second opcode node.

26. (Previously Presented) The one or more computer-readable storage media as recited in claim 24, the executing opcode nodes further comprising executing each opcode node a single time.

27. (Previously Presented) The one or more computer-readable storage media as recited in claim 24, the queries further comprising XPath queries.

28. (Previously Presented) The one or more computer-readable storage media as recited in claim 24, further comprising executing one or more branch nodes to execute one or more opcode nodes that depend from the branch node.

29. (Previously Presented) The one or more computer-readable storage media as recited in claim 28, the branch node further comprising an indexed lookup procedure to optimize execution of the dependent objects.

30. (Previously Presented) The one or more computer-readable storage media as recited in claim 24, further comprising executing one or more indexed branch nodes to

execute a plurality of opcode nodes that depend from the branch node, the plurality of opcode nodes including a similar comparison function.

31. (Previously Presented) The one or more computer-readable storage media as recited in claim 30, the indexed branch node including a hash function, the opcode nodes including a literal comparison function.

32. (Previously Presented) The one or more computer-readable storage media as recited in claim 30, the indexed branch node including an index lookup procedure selected from the following list of index lookup procedures: a hash procedure; an interval tree procedure; a procedure utilizing tries.

33. (Previously Presented) The one or more computer-readable storage media as recited in claim 24, further comprising receiving an input that is evaluated against the plurality of queries using the opcode tree.

34. (Previously Presented) The one or more computer-readable storage media as recited in claim 33, further comprising a compiler configured to execute each query in the plurality of queries to derive the opcode nodes.

35. (Previously Presented) A method, comprising:

evaluating input against multiple queries by evaluating common query expressions of the multiple queries in parallel at a same time;

merging opcodes into an opcode tree, wherein elemental language units have been parsed and compiled into opcodes, wherein no opcodes are added to the opcode tree during an active merging;

executing an opcode tree of hierarchical nature that includes opcode nodes as encountered that each correspond to one or more of a plurality of XPath queries represented by the opcode nodes, at least a first opcode node corresponding to a first query and a second query;

indexing branch opcodes to provide a framework for insertion of indexing techniques that are customized to the type of comparison;

using interim values from an execution context created in the execution of the first opcode node in the execution of a second opcode node corresponding to the second query to avoid re-creating at least a portion of the execution context;

maintaining the opcode tree that is used during processing by making a copy of the opcode tree; and

updating the opcode tree copy, wherein the opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing;

wherein a relationship between the opcodes and the opcode trees is embedded in the opcodes that is created when a query is complied.

36. (Original) The method as recited in claim 35, the executing the opcode tree further comprising executing at least one branch node in order to execute two or more opcode nodes that depend from the branch node.

37. (Original) The method as recited in claim 36, the branch node further comprising an index lookup function to optimize execution of the two or more dependent opcode nodes.

38. (Original) The method as recited in claim 37, the indexed lookup function further comprising a hash function, a function utilizing tries or an interval tree function.